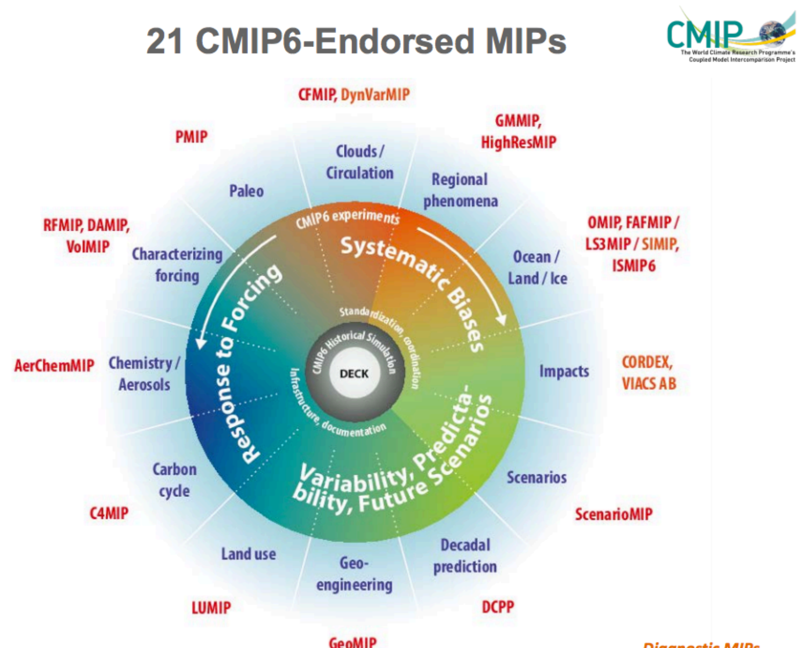
Using Parallel Python Tools to Postprocess Data for CMIP6

Sheri Mickelson
Kevin Paul

Eighth Symposium on Advances in Modeling and Analysis Using Python
AMS 2018

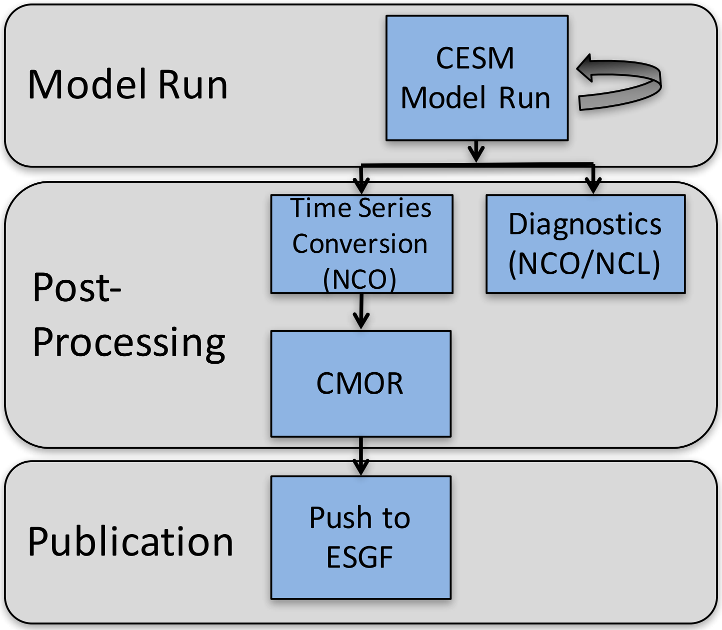
What is CMIP6?

- Internationally coordinated effort to run sets of defined experiments. There are roughly 30 different centers from around the world that will be participating.
- Each experiment has a defined set of protocols, forcings, and requested output.
- Running the same experiments with multiple models leads to stronger results.
- The results of these simulations are evaluated as part of the IPCC Climate Assessment Reports. The results are also used by international governments for policy decisions and for further research by scientific institutions and universities.



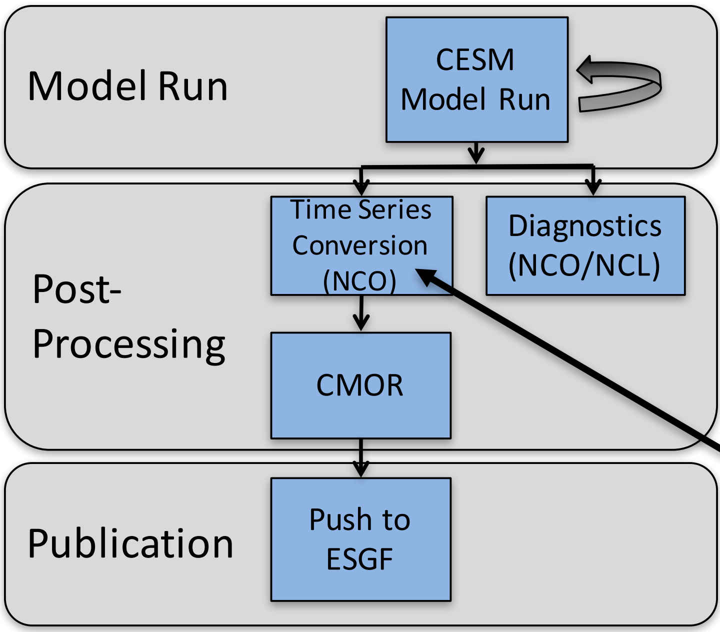
Eyring et al., GMD, 2016

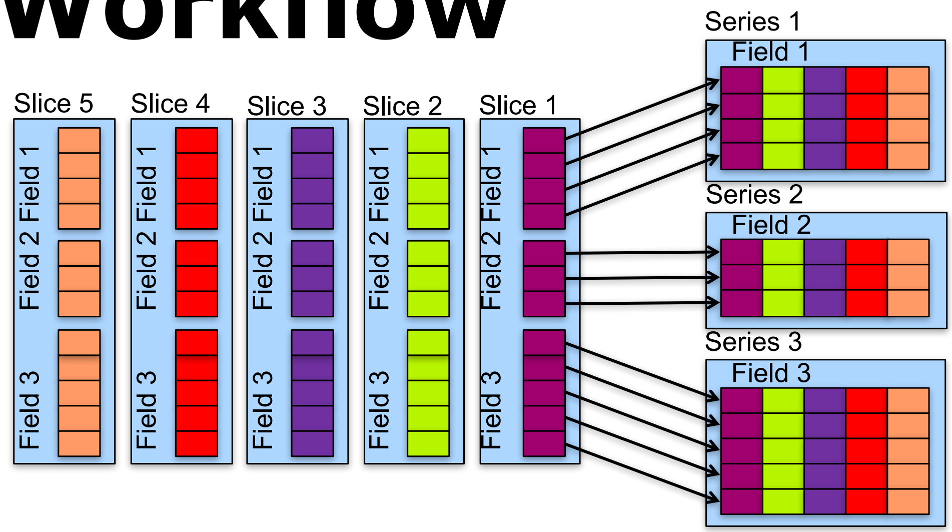
CMIP5 Workflow



These are all of the steps that we need to take to publish our data to the community. For CMIP5 this process took 15 months to postprocess 200 TB of data.

CMIP5 Workflow

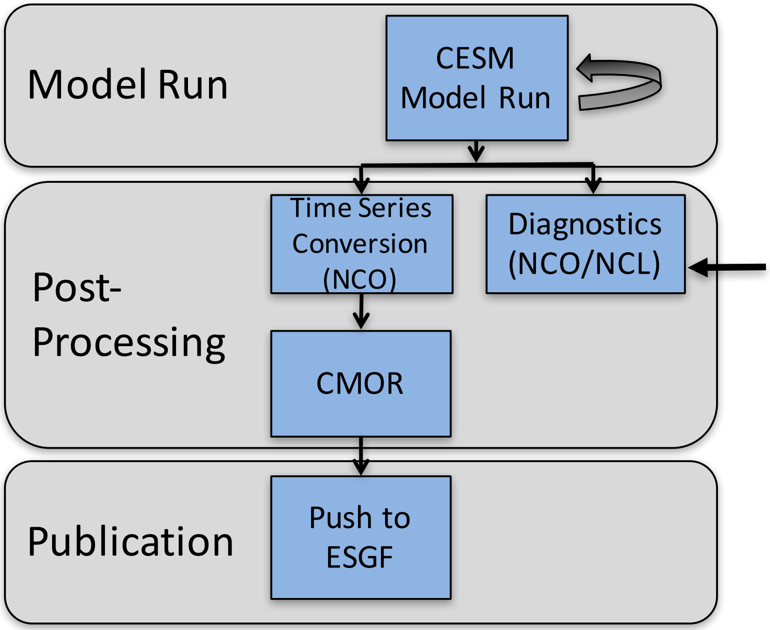




Converting from Time Slice to Time Series

Model outputs data in a format that has one time slice and multiple variables. The preferred distribution format is one variable and multiple time slices. This step converts from one format to the other. The existing method used NCO for the conversion. This was the most expensive step in CMIP5.

CMIP5 Workflow



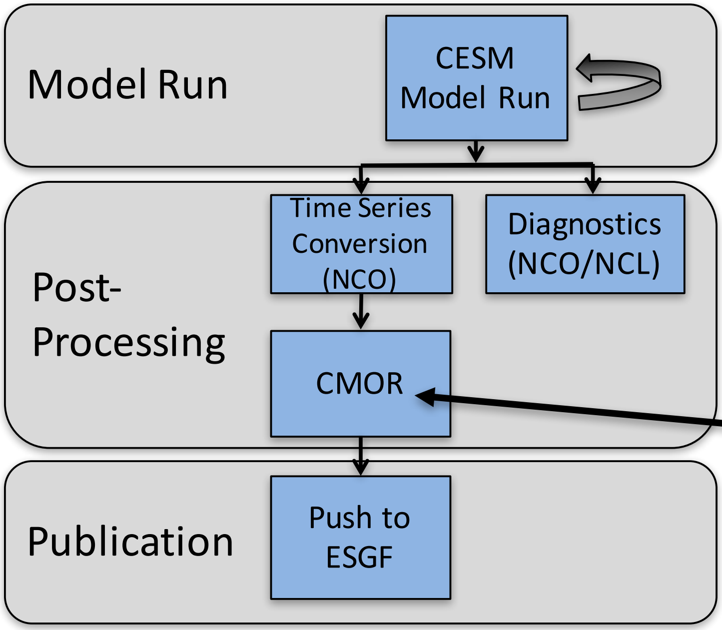
6 Component Diagnostic Packages
(Atm, Lndx2, Ocn, SeaIce, BGC)

Used to document and evaluate the climate simulation.

All of the original packages:

1. Contain a top level control script
2. Create climatology files with NCO tools
3. Create hundreds of plots with NCL scripts
4. Create web pages that allow users to browse through plots

CMIP5 Workflow



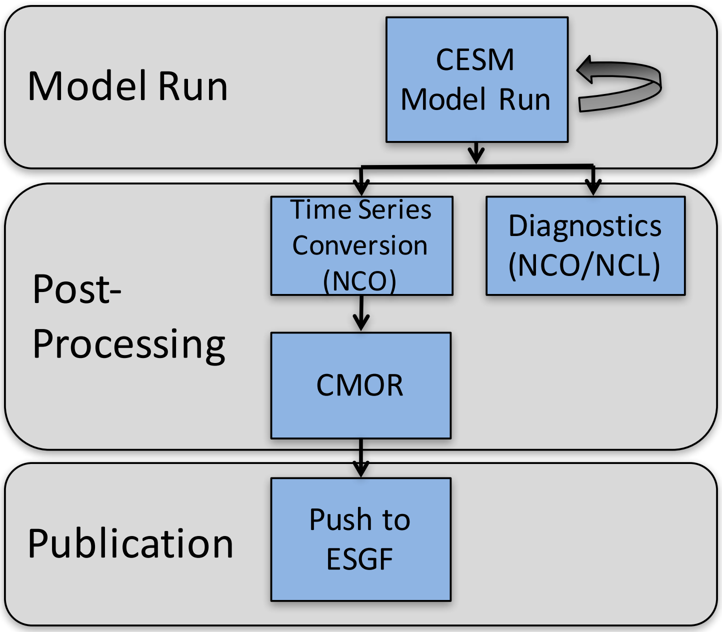
This step **Standardizes** the model output in in order to make it easier to compare against other models for the intercomparison.

Some examples:

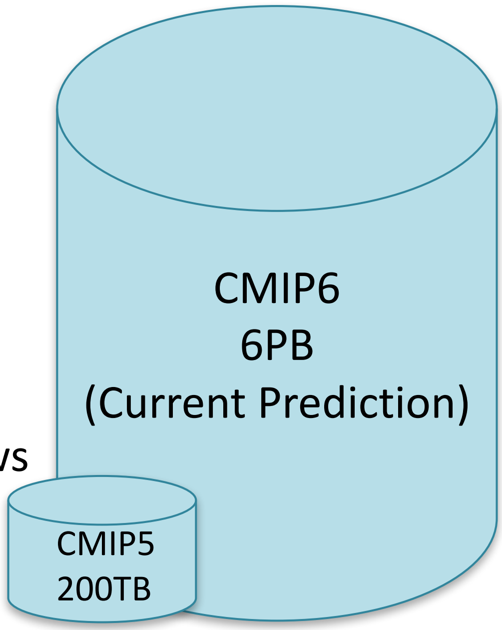
- File formats (*e.g.*, NetCDF4)
- Names of files and directory structure
- File attributes (e.g., institution, MIP name, …)
- Names of dimensions (*e.g.*, `lat`, `lon`, …)
- Names of variables (*e.g.*, `psl`, `ta`, `tas`, …)
- Dimensions of variables
- Variable data types (e.g., `float`, `double`, …)
- Attributes of variables (*e.g.*, `units`, …)
- Ranges of time (*e.g.*, 2006 to 2100)
- Deriving variables that are not outputted directly

Used Fortran and NCL code, NCO, and CMOR

CMIP5 Workflow



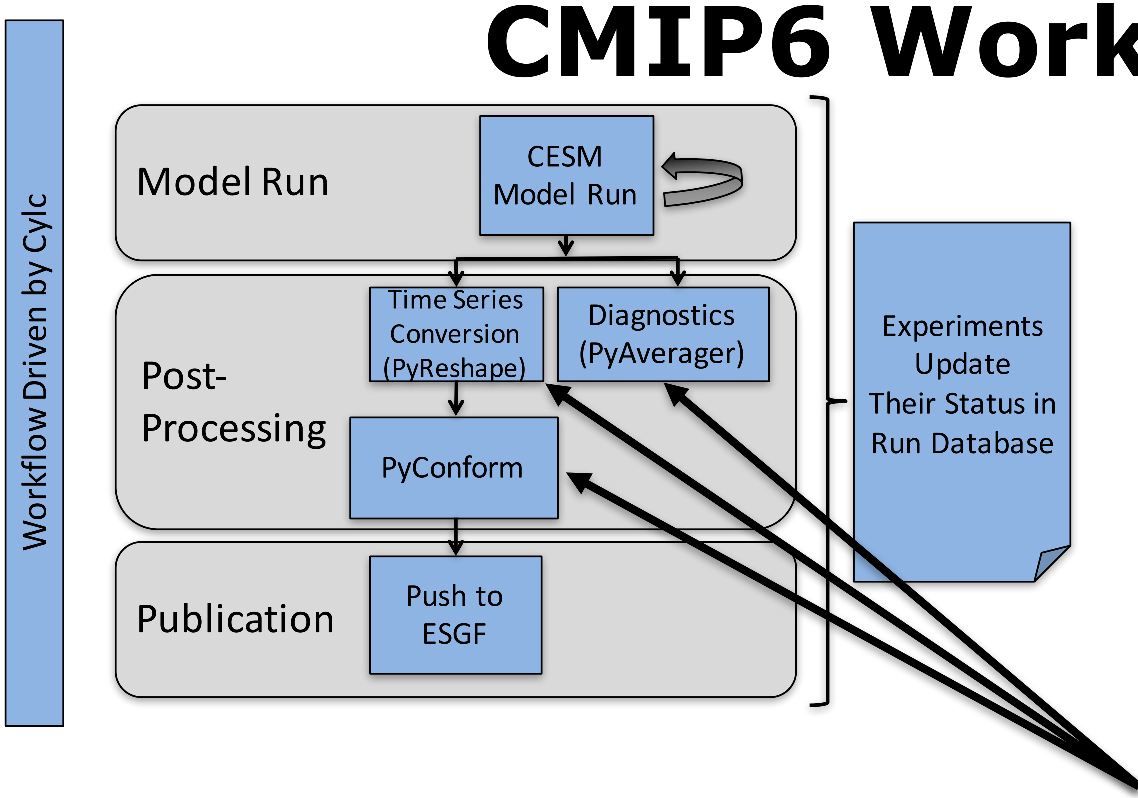
Motivation:
For CMIP6 we will have to postprocess 6 PB of data within the same amount of time. We needed better methods in order to be able to create this amount of data for the community in time for AR6.



We Needed to do Three Things:

1. Increase Performance: Added parallelization into the workflow
2. Reduce Human Intervention: Worked on integrating our workflows into an automated workflow engine
3. Project Management: Everything is coordinated through a central database

CMIP6 Workflow

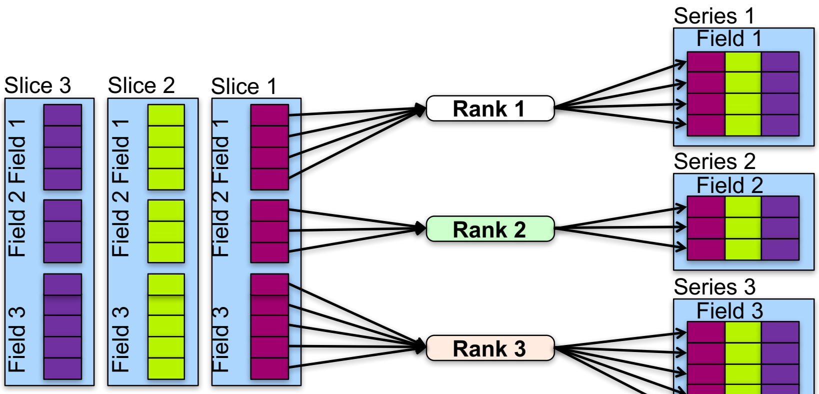


We rewrote tools in Python and added task parallelization.

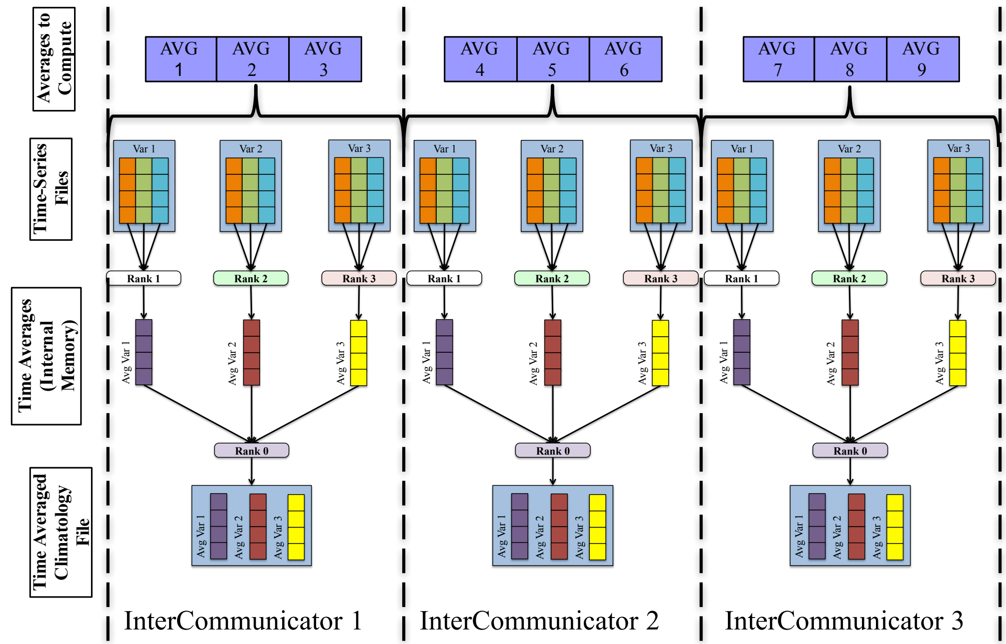
All of the tools depend on:

- MPI4Py for internode communication
- PyNIO and NetCDF4-python for I/O

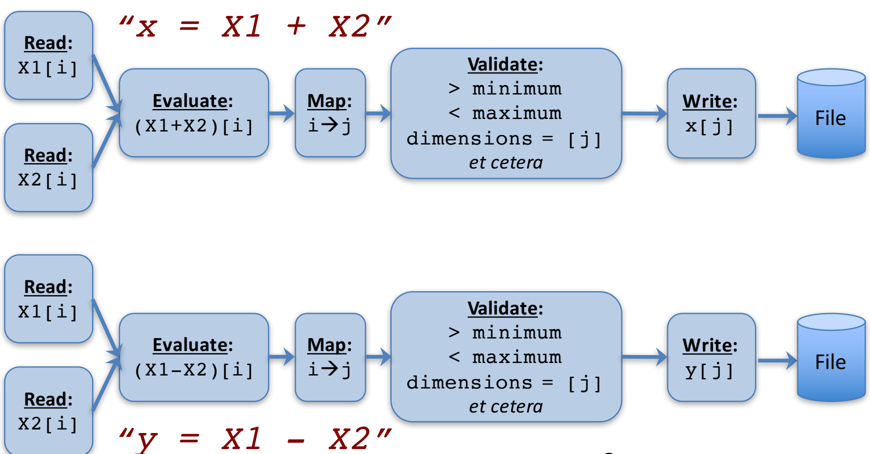
Parallelization Methods



PyReshaper

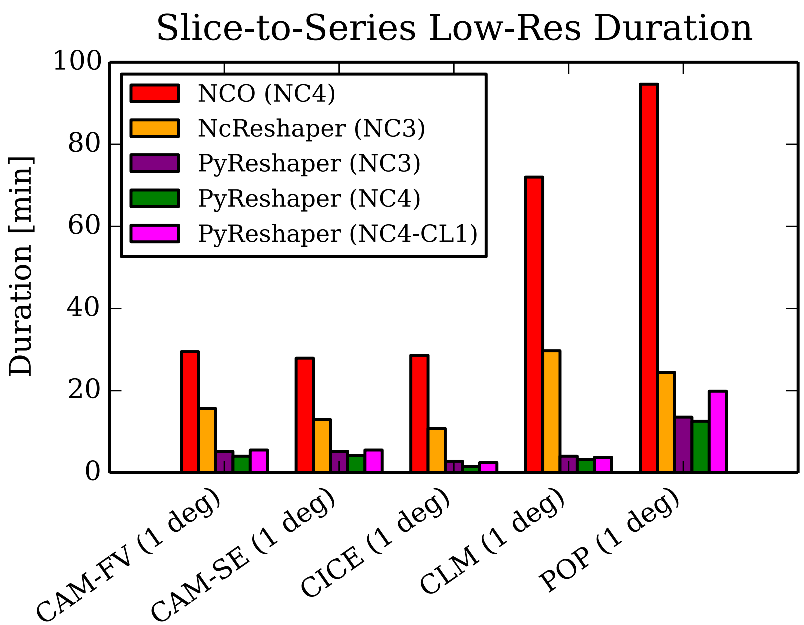


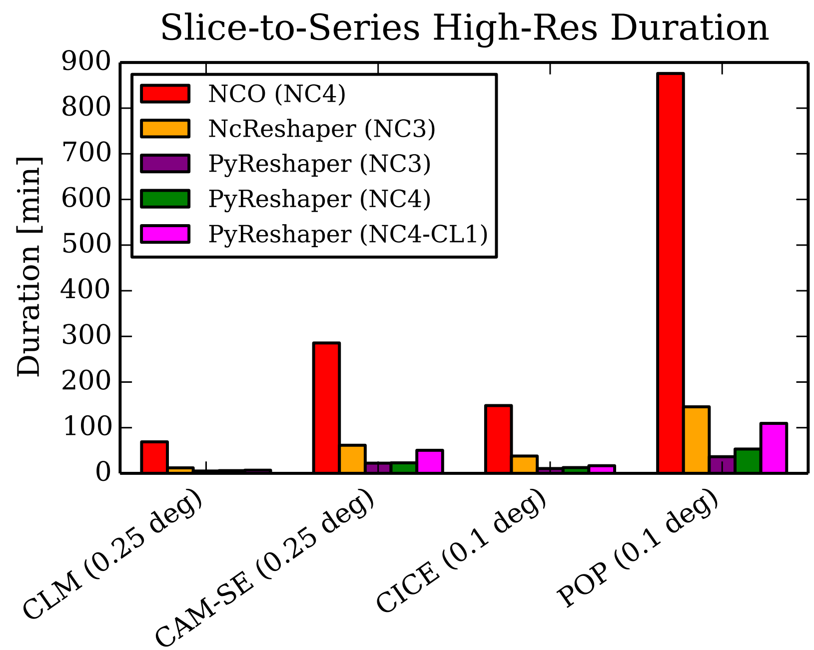
PyAverager



PyConform

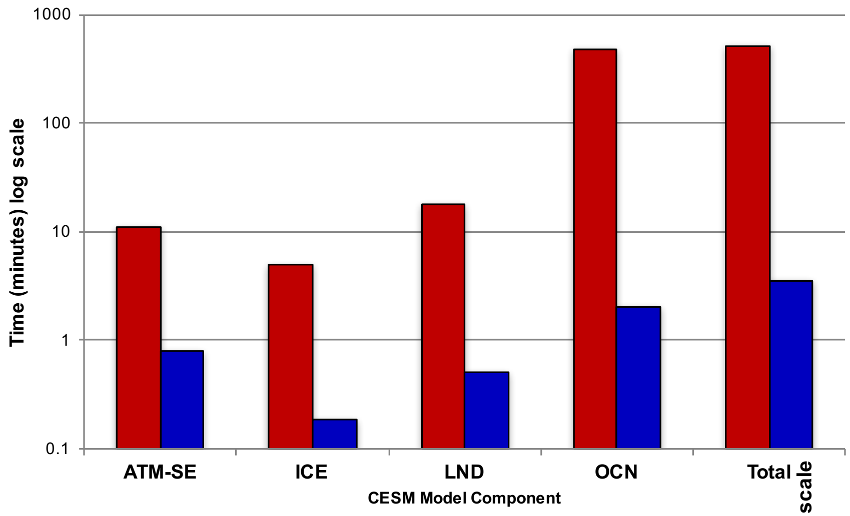
PyReshaper Performance



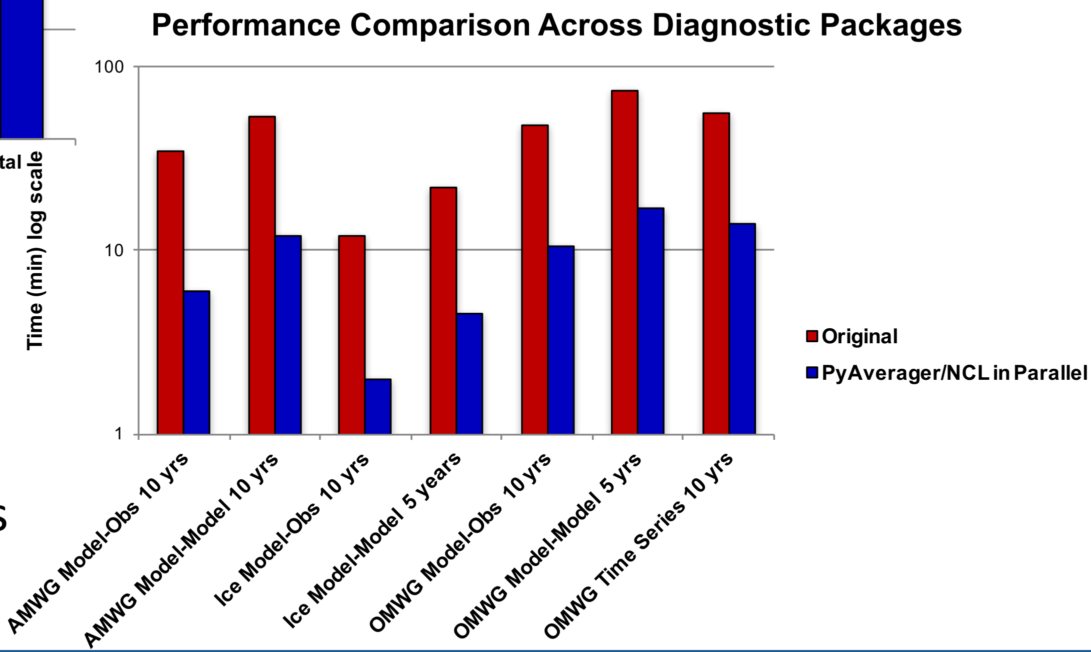


Results are from running the PyReshaper tool on 16 yellowstone cores, 4 cores on 4 nodes

PyAverager Performance



Time to compute climatology files for 10 years of CESM monthly time slice files.
The PyAverager ran on 120 cores on yellowstone and the diagnostics on 16 yellowstone cores.

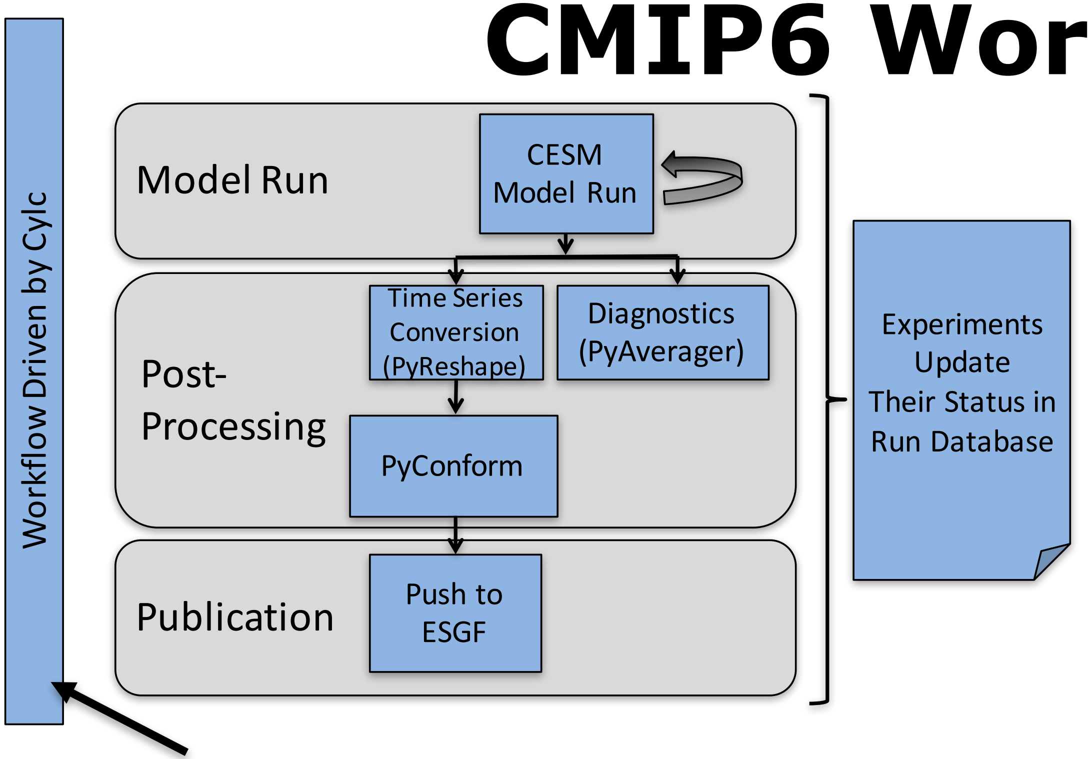


PyConform Performance

(Preliminary Timing Numbers)

CESM Case Name	CMIP5 Table	Input Dataset Size	Output Dataset Size	Original Serial Runtime	PyConform Parallel Runtime (16 Procs)	SPEEDUP
b40.rcp4_5.1deg.006	Amon	84 GB	62 GB	72 mins	2 mins	**38x**
b40.20th.track1.1deg.012	Amon	135 GB	102 GB	120 mins	8 mins	**16x**
	3hr	540 GB	506 GB	6 hours	11 mins	**34x**

CMIP6 Workflow



We adopted Cylc as our workflow Engine (written by Hilary Oliver at NIWA)

We auto-generate the workflow description files from both the CESM and postprocess environments. All the user needs to do is edit a top level script to set certain variables (run length, which diagnostics to run, etc) and then manually start the run through Cylc's GUI or command line interface.

Use Cases of Experiments That Used Cylc

- Used Cylc to complete 1,240 out of 1,860 total runs ~750 TB timeslice output in about 1 month
- Used Cylc to run and postprocess part of a 30 member ensemble in a couple of months
- Used Cylc to build and run over 20,000 forecast ensembles in a couple of months

Questions?

- PyReshaper
  - https://github.com/NCAR/pyreshaper
- PyAverager
  - https://github.com/NCAR/pyAverager
- PyConform (still in development)
  - https://github.com/NCAR/PyConform
- CESM/Cylc WF
  - https://github.com/NCAR/CESM-WF
- Cylc
  - https://cylc.github.io/cylc/

Conact Info mickelso .at. ucar.edu